# Keyword Analysis and Automatic Summarization

NLP methods for analysis of beauty product reviews and focus group responses


## Abstract

Text data provide an abundant source of rich yet unstructured information. For the corporate user, text data is crucial for understanding customer sentiment. We present an overview and demonstration of methods for keyword extraction and text analysis, including topic modeling. Lastly, we implement two unsupervised methods for multi-document extractive summarization. Our implementation of the latter, Yogatama et al's semantic volume maximization (2015), is to our knowledge the first implementation aside from the original (which is not publicly available).



Joshua Safyan
jds2258@columbia.edu
Shengzhong Yin
sy2615@columbia.edu
Junhui Liao
jl4574@columbia.edu
Shuni Fang
sf2599@columbia.edu

## I. Introduction [by Joshua Safyan]

Since the advent of the printing press in the 15th century, text has played a crucial role in knowledge transmission. To meet the challenge of a world of rapidly increasing data, our means of comprehending it must scale as well. With finite time, one must find an automated way to extract key information from one or more documents. While the internet has accelerated its importance, automated summarization has received considerable attention from the academic community for the majority of the modern computer age. Beginning in the late 1950s with Luhn (1958)[1] who used term frequency heuristics on an IBM 704 to summarize text, researchers have since discovered a multitude of techniques and strategies to condense text content.

It is worth explicitly stating at the onset what constitutes a summary. We posit that a summarization process produces abbreviated output from one or more documents, such that salient information from the source documents is preserved[2]. While this is hardly a formal definition, for a summary to be useful to humans it must be 1) considerably shorter than the input source and 2) retain key information[3]. Broadly, summarization takes two forms -- *abstractive summarization* and *extractive summarization*[4]. For the former, the summarizer rephrases input content in order to compress the information, whereas the latter framework identifies and presents representative excerpts from the source material.

## II. Goals [by Joshua Safyan]

We aim to provide automated summarization of beauty product reviews by identifying key information, themes, and sentiment. Unilever, the project mentor, seeks to research the feasibility of such information extraction for their unstructured text data from beauty product reviews and focus groups. To that end, we were given access to a similar dataset of Amazon beauty product reviews to investigate potential approaches.

More specifically, we are interested in multi-document summarization (rather than single-document)[5]. Abstractive summarization provides the benefits of a more natural fusion of concepts from disparate sources without templating. Additionally, it allows more flexibility through rephrasing. However, modern abstractive models typically use deep learning, which can be computationally expensive and often requires large datasets. As a result, extractive summarization is often considerably easier to rapidly prototype. Thus, our initial approach uses

---

[1] http://courses.ischool.berkeley.edu/i256/f06/papers/luhn58.pdf
[2] Das and Martins (2007) http://www.cs.cmu.edu/~nasmith/LS2/das-martins.07.pdf
[3] Ibid.
[4] Ibid., Nenokova and McKeown https://www.cs.bgu.ac.il/~elhadad/nlp16/nenkova-mckeown.pdf
[5] According to Das and Martins, the Columbia NLP group (McKeown and Radev, 1995; Radev and McKeown, 1998) pioneered multi-document summarization in the mid-1990s with a system called SUMMONS that combined template-based text understanding and fusion
Das and Martins (2007) http://www.cs.cmu.edu/~nasmith/LS2/das-martins.07.pdf

extractive methods with a stretch goal of an abstractive proof of concept for the end of the semester.

Extractive summarizers may contain multiple stages and methods combined together to give a holistic view of the data. Prior work includes a number of diverse approaches. Often, methods are employed to identify the relative importance of source data, whether bag of words counts, tf-idf, or a graph-based ranking system like LexRank[6] or PageRank for co-occurrence of words. Modern methods include topic modelling (e.g., using LDA[7]), embeddings[8], and others. Embeddings are of particular interest because they allow one to use word, sentence, or paragraph context to inform meaning, such that one can compare concepts in vector space that have otherwise dissimilar natural language string representations.

## III. The data [by Joshua Safyan and Shuni Fang]

**Amazon Beauty Product Review Dataset**
The Amazon beauty product review dataset consists of 198,462 reviews written from June 2002 to July 2014, spanning 12,075 unique products written by 22,363 unique reviewers[9]. Along with the review text, the data contains a number of metadata fields including the review summary (essentially the title of the review written by the reviewer), the date, the product id, the reviewer id, the rating (on a scale of 1 to 5), and the numbers of people who found the review helpful and not helpful.

**Unilever Focus Group Dataset**
The Unilever focus group dataset contains about 450 responses to free response questions and multiple choice questions. It consists of eight columns in question-and-answer format: each column contains survey answers to the column question, such as “What is Healthy Skin?”, “How do you know your skin is healthy?”. The answers to the questions are short and to-the-point. Because of the Q&A nature of the data, the text is much cleaner and focused than the text in the amazon product review dataset.

## IV. Initial Exploration [by Shuni Fang]

There are two approaches to automatic summarization: extractive and abstractive. Extractive summarization creates summary by concatenating extracts of existing words, phrases, sentences from a corpus and rearranging them into a summary.

---

[6] http://www.jair.org/media/1523/live-1523-2354-jair.pdf
[7] Blei et al (2003) http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
[8] https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
[9] **Inferring networks of substitutable and complementary products** J. McAuley, R. Pandey, J. Leskovec *Knowledge Discovery and Data Mining*, 2015 pdf

It creating summary based strictly on what you get in the text. it can be compared by copying down the main points of a text without modification to those points and rearranging the order of that points and the grammar to make more sense of the summary.

In contrast, abstractive summarization is paraphrasing the corpus, which can produce more condensed summaries and makes the summary look more human-like.

While abstractive summarization sounds more attractive to our purpose than extractive summarization, it also entails higher complexity. Given that our data contains titles of the reviews and does not contain text summaries , we currently focus on extractive summarization. In particular, we currently focus on k-means clustering of document vectors for summarization as well as topic modeling.

## V. Text Rank for extracting keywords [by Shuni Fang]

By using text rank instead of just counting the total number of occurrences of words in the texts, we were able to get a list of words that are more relevant to what we want from the product reviews. Below is the top 50 words in the the reviews using text rank (sorted in descending order):

'skin', 'product', 'face', 'good', 'great', 'hair', 'year', 'lotion', 'time', 'cream', 'hand', 'day', 'dry', 'acne', 'little', 'eye', 'moisturizer', 'smell', 'price', 'soft', 'oil', 'makeup', 'oily', 'best', 'sensitive', 'color', 'night', 'week', 'light', 'long', 'nice', 'cleanser', 'shampoo', 'bit', 'clean', 'greasy', 'way', 'brand', 'bottle', 'stuff', 'better', 'thing', 'perfect', 'fragrance', 'difference', 'smooth', 'iron', 'month', 'hard', 'sure'

TextRank works well it takes into account information recursively drawn from the entire text (graph). Through the graphs it builds on texts - it identifies connections between various entities in a text, and implements the concept of recommendation. [10]

If we combine this technique with facet identification (explained down below), as well as bi-grams and tri-grams, we would be able to get the key words and phrases associate with each facet of the products. That is one of our goals in the next stage.

## VI. Distributed Representations of Sentences and Documents [by Junhui Liao]

[10] Rada Mihalcea and Paul Tarau, https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

Nowadays, most of the machine learning techniques require features as numeric vector so we find it reasonable to represent reviews as vector before we apply different models to summarize these product reviews. Such technique was first published in Distributed Representations of Sentences and Documents (Le  & Mikolov et al., 2014). In addition, two models are proposed in their paper.

## Model 1 Paragraph vector: A distributed memory model (PV-DM)

Each paragraph (it can be either a sentence or sentences) is projected to be an unique column vector representation in matrix D. In addition, every word is also transformed into a unique column vector representation in matrix W. In order to predict the next word in a context, concatenation or average will be utilized to paragraph vector and word vectors. Please see figure 1, which is from the original paper ( Le  & Mikolov et al., 2014), for the framework.

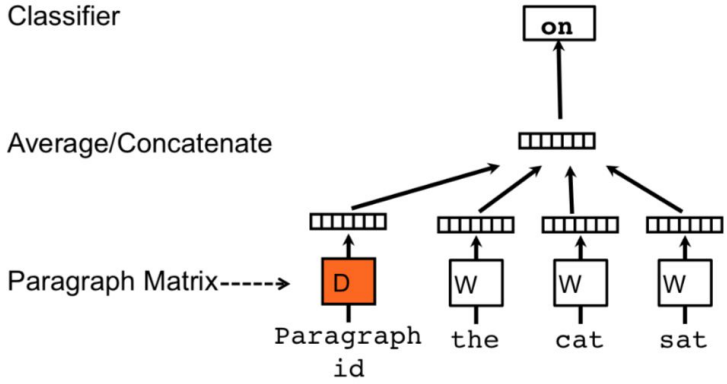



Figure 1. A framework for learning paragraph vector.

## Model 2 Paragraph vector without word ordering: Distributed bag of words (PV-DBOW)

This model disregards the context words as input but predict words randomly sampled from the paragraph as output. In other words, during each iteration, we sample a text window and then, sample a random word from the text window as output. Given a paragraph vector, we form a classification task to predict such output. Please see figure 2 for framework (Le & Mikolov et al., 2014).

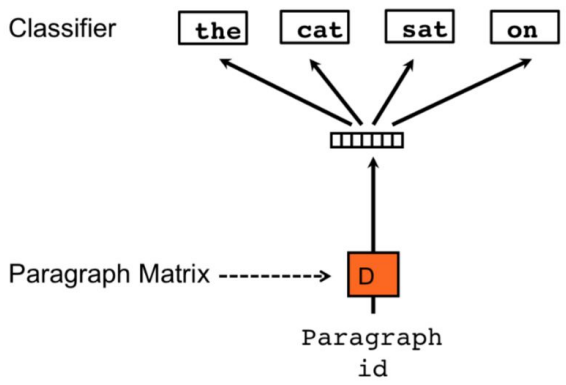



Figure 2. Distributed Bang of Words version of paragraph vectors.

Experiments conducted by Le and Mikolov shows PV-DM works well for most tasks, however, with PV-DBOW combined they can achieve more consistent result across many tasks. Therefore, it is highly recommended. Base on this finding, we decide to utilize both PV-DM and PV-DBOW vectors for future summarization task.

## VII. k-means clustering for extracting representative reviews [by Joshua Safyan]

To find a small number of reviews that represent the corpus of reviews for a given product, we first create document vectors for each review. Following Le and Mikolov (2014)[11], we create 400 dimensional vectors for each review. Using an embedding of the paragraph rather than bag of words approaches allows for incorporation of contextual information. Additionally, it allows for different phrasings of similar concepts to be embedded close to one another in vector space.

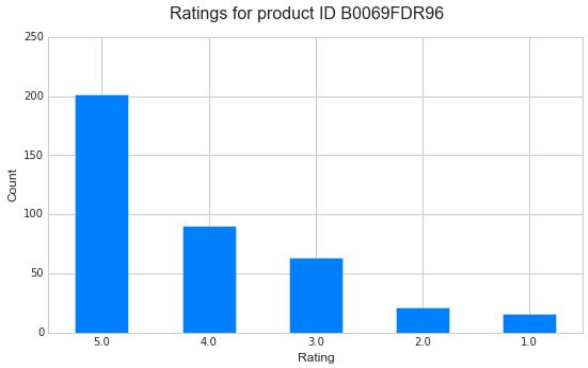
Once these review vectors have been obtained, k-means (with k-means++ initialization) is used to find cluster centroids. The number of clusters is determined by cross validation using the silhouette score with a cosine distance metric, which ranges from -1 to 1 (where -1 corresponds to overlapping clusters, and +1 indicates cluster separation). Given the number of clusters, k-means is run on the document vectors for a given beauty product's reviews, yielding the centroid vectors.

Given the centroid vectors, we can now find representative reviews for a product. The first method explored used cosine distance from the centroids to find the nearest document vectors, and then returned the corresponding review text for those vectors. While we plan use metrics such as ROUGE to evaluate our automatic summarization in the future, on this initial investigation we used other heuristics to evaluate the quality of the summaries.

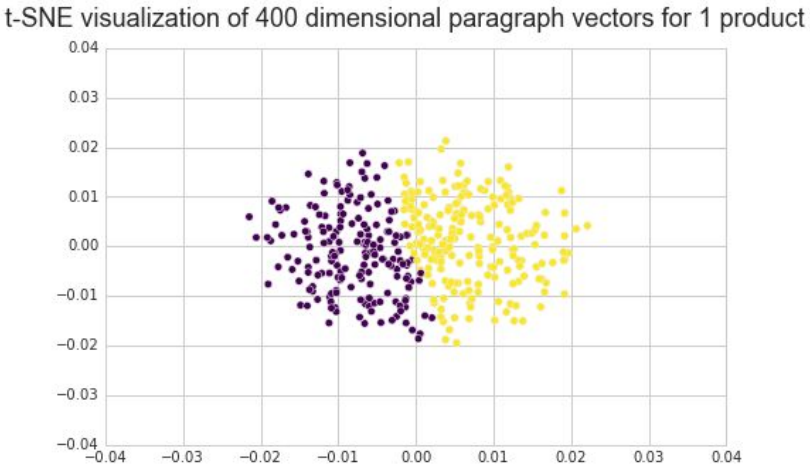
[11] Le, Mikolov (2014) http://cs.stanford.edu/~quocle/paragraph_vector.pdf

## Example: product ID B0069FDR96 [by Joshua Safyan]

This product features 391 reviews, which we seek to summarize. To gain an idea for the overall sentiment about the product, we examine the distribution of overall ratings. Roughly three-quarters of the ratings are favorable, but upon cursory inspection it was found that people may be kind in their ratings -- review texts that sounded fairly mixed sometimes gave the products 4 out of 5 stars. Clearly this observation is anecdotal, but it suggests that it may be a good idea to train a classifier to first classify reviews as {positive, negative} and then summarize information from the positive and negative classes separately to increase the diversity of opinions.




Performing a grid search on k-means cluster sizes, we find that the optimal cluster size is 2 with silhouette score ~0.1, which indicates slight cluster separation. We can visualize the 400-dimensional document vector clusters by embedding them in lower dimension space using t-SNE with PCA initialization. One cluster has 181 reviews, while the other has 210 reviews.




Using cosine distance, we find that the document vectors closest to each centroid are:

> *Packaged individually,easy to start pull tab,sticky tape stays on until you pull it off. Small price to see if you enjoy doing this.*
>
> *These took forever to come,but they are worth the wait. They are very much like the picture. A good value I will order again if the need arises.*

Unfortunately, both of these reviews are fairly positive but we know from the distribution of ratings that there should be a non-negligible contingent of mediocre to negative reviews. This was foreshadowed by the low silhouette score, since scores near 0 indicate that the clusters aren't very well separated. We are interested in varied summaries, so we need a means of adding richness to the reviews we return. One method to explore as mentioned above may be to first classify reviews as positive or negative and then surface representative reviews from each group. Additionally, topic modeling (whether on all reviews for a product or segmented by positive and negative reviews) may prove useful for ascertaining the distribution of sentiment and themes.

To combat the problem of the two centroid reviews being too similar, we used a similarity score built on top of the cosine distance. To create this score, one specifies positive ("like") document vectors and negative ("unlike") document vectors. The cosine distance for "like" vectors adds to the similarity score, while the cosine distance for "unlike" vectors detracts from the similarity score. We use this metric for each cluster, specifying its own centroid as "like", and all other centroids as "unlike," maximizing its likeness to its own cluster while minimizing its similarity to unlike clusters. This allows us to obtain vectors that are more characteristic of their cluster, conditional on all other clusters.

Using this method on the same product reviews, we obtain the following:

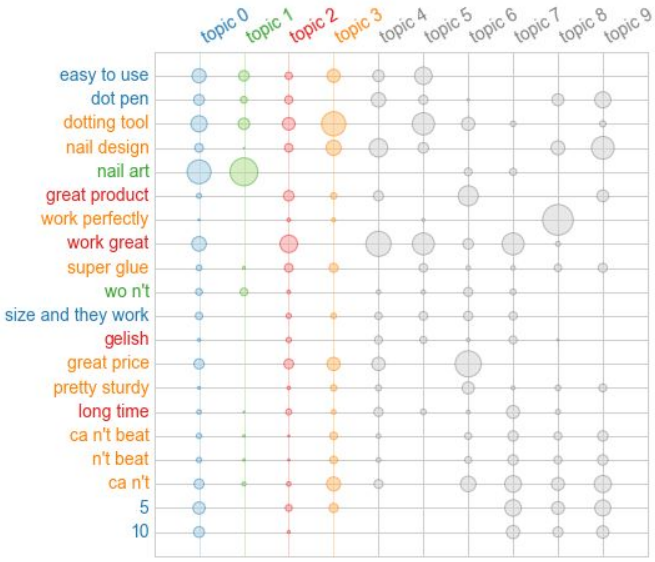
> *For starters,the palette is a lot smaller than it appears. The pigmentation of the concealers is terrible and they feel very oily before you deposit them on to the skin. Once applied,they are incredibly chalky and leave the skin feeling dry and dehydrated. I would not recommend this item to a friend.*
>
> *These took forever to come,but they are worth the wait. They are very much like the picture. A good value I will order again if the need arises.*

Now we have a negative review for the minority class and a positive review (the same as the previous positive review) for the majority class. This aligns better with our expectations, but may give the incorrect impression that nearly half of the reviews are negative, so it is a bit of an overcorrection. As evidenced by this example, incorporating proportional richness and diversity into the summaries will be essential to gaining an accurate depiction of the reviews for a product.

Other avenues for exploration include adjusting the pre-processing pipeline -- instead of using a document vector for an entire review, each review could be further subdivided into sentences with each sentence vectorized. This could potentially also allow for better separation of concepts. If many reviews are mixed, the resulting document vectors for the reviews will be close together even when the individual sentence concepts may differ significantly.

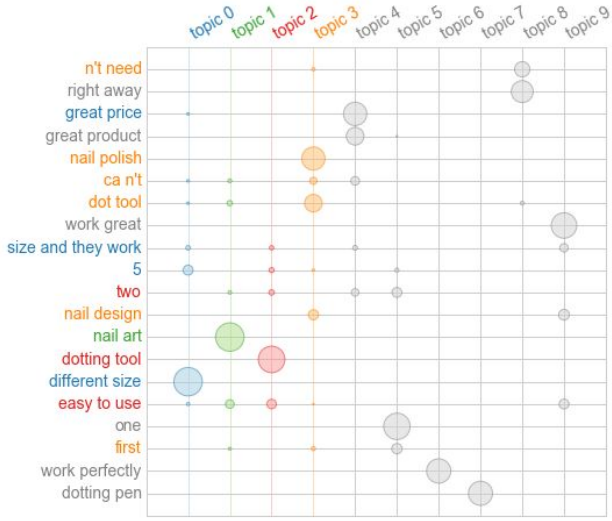
## VIII. Topic Modeling [by Joshua Safyan]

We began to examine three different approaches to topic modeling, each of which has strengths and weaknesses[12]. The first two, Latent Semantic Analysis (LSA) and Non-Negative Matrix Factorization (NMF), both use somewhat similar matrix decompositions of document-term matrices to identify low-rank approximations of the original document-term matrix. Using the example product id above, we created a smoothed tf-idf document-term matrix of lemmatized (2, 3, 4)-grams and filtered based on a minimum count of 2 in the corpus and a maximum frequency of appearing in 95% of the corpus documents. Below, we can see representative terms for the top 10 topics for LSA and NMF.



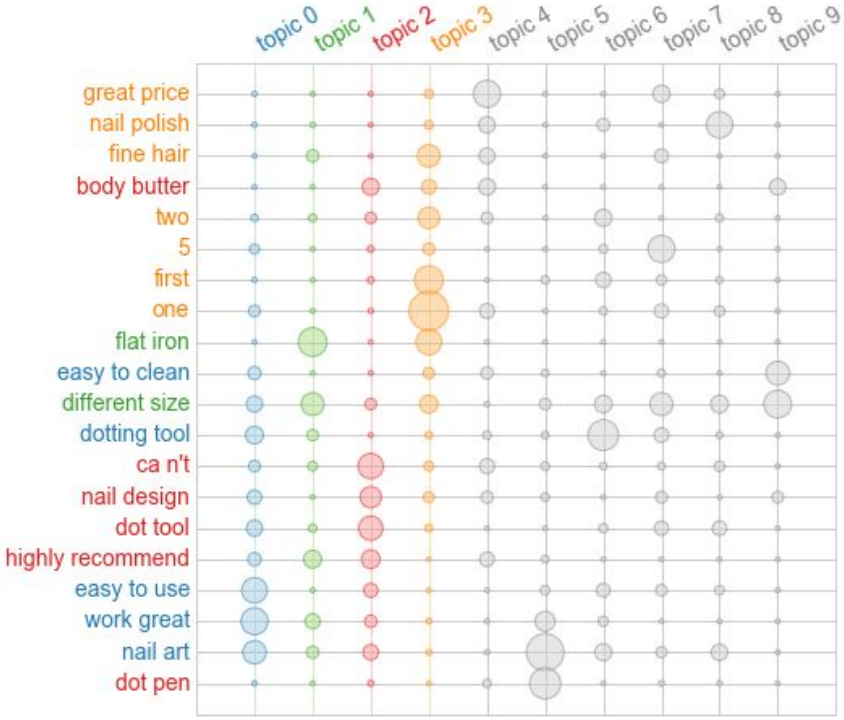

LSA

[12]Stevens et al (2012) http://aclweb.org/anthology/D/D12/D12-1087.pdf




NMF

Lastly, we used the document-term matrix to construct a Latent Dirichlet Allocation (LDA)[13] model and visualize the top 20 terms for each of the 10 topics.




LDA

LDA tends to produce more coherent, human understandable topics[14], and has added flexibility due to the Dirichlet prior on the topic distribution for a given document. We plan to explore topic models further and their applicability to product review summarization.

[13] Blei et al (2003) http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
[14] Stevens et al (2012) http://aclweb.org/anthology/D/D12/D12-1087.pdf

## IX. Exploratory Sentiment Analysis on Amazon dataset [by Shuni Fang]

Given that the amazon dataset is free text on different beauty products, we want to learn what the users feel about the product. So we conducted lexicon-based sentiment analysis for average emotion valence. The picture below shows the average emotional valence on the whole amazon product review corpus:

```
defaultdict(float,
            {'AFRAID': 4.906505396191408e-07,
             'AMUSED': 6.842031932680173e-07,
             'ANGRY': 4.6116122255680835e-07,
             'ANNOYED': 5.332288625423938e-07,
             'DONT_CARE': 6.126061042950041e-07,
             'HAPPY': 5.723068900022761e-07,
             'INSPIRED': 6.313005015063396e-07,
             'SAD': 5.13162316054975e-07})
```

Since we didn't know what kind of product (ie. product category) a particular review correspond to from the dataset given, we were not able associate the emotions with each product. But it is something we can do in the future: associate emotional valence with specific product and product facets.

The sentiment analysis does not apply on the Unilever focus group dataset because the texts in the Unilever dataset are answers to "factual" questions, ie. the survey participants are not expressing how they feel about any products.

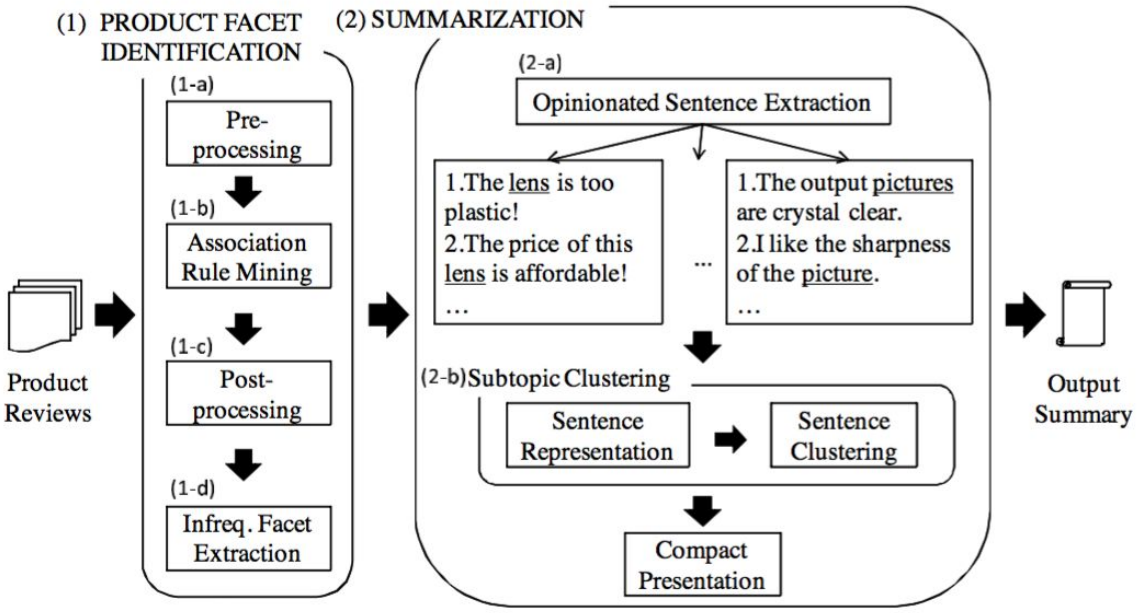
## X. Facet Identification [by Shuni Fang]

In the domain of product reviews, finding the orientation of the sentence is generally not enough. it is necessary to identify the semantics of the opinion in the sentence, as the opinion holder may describe a particular facet of the subject in the review that users may be interested in.

The beauty product reviews are generally divided into three categories: skincare, haircare, and color cosmetics. For our purpose of extracting useful information from the product reviews, finding the orientation of the sentence and a general summary are not enough. We want to learn the particular aspects that concern about a beauty product. As a result, we need to do product facet identification. Typical examples of facets that belong to a beauty product would be: the product itself, the packaging, the retail environment, the website's online shopping experience,

among which we are mostly concerned about the product itself. To identify the facets, we plan on utilizing WordNet - a large lexical database of English where nouns, verbs, adjectives, adverbs are grouped into sets of cognitive synonyms, each expressing a distinct concept - to grow a initial seed list of known orientation adjectives into a larger list that covers all the remaining adjectives in WordNet. We choose this system because of its strong sense of organization compared with use of large text or Web corpora. [15]

The paper “Product Review Summarization based on Facet Identification and Sentence Clustering” proposes a system that combines sentiment analysis and text summarization with WordNet: [16]




Given that the reviews we are summarizing is product specific, it makes sense for us to try this approach in the next stage.

## XI. Keyword extraction - two methods comparison [by Shuni Fang]

To get the top words from each column, we tried two methods: 1. counting the number of occurrences of each word; 2. Using textrank algorithm[17] to return the most important and relevant words.

---

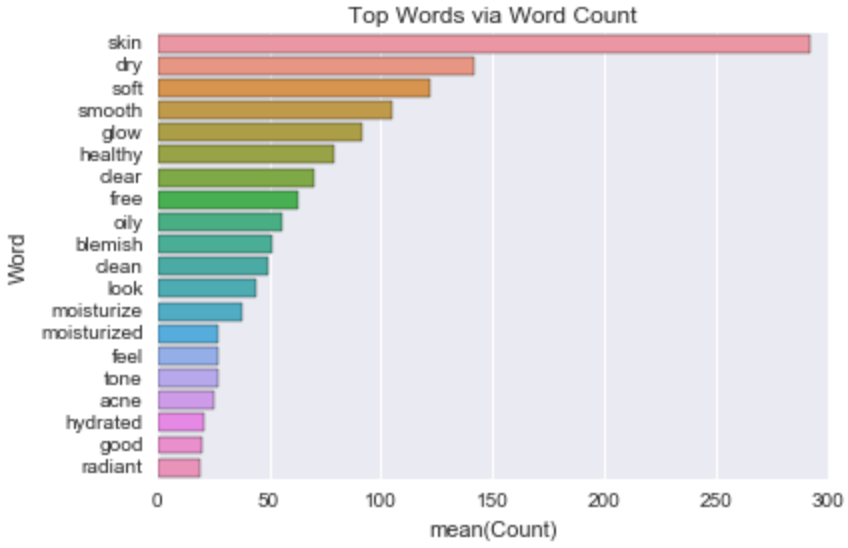
[15] Duy Khang Ly, Kazunari Sugiyama, Ziheng Lin, Min-Yen Kan (2011) https://arxiv.org/pdf/1110.1428v1.pdf
[16] Duy Khang Ly, Kazunari Sugiyama, Ziheng Lin, Min-Yen Kan (2011) https://arxiv.org/pdf/1110.1428v1.pdf
[17] https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

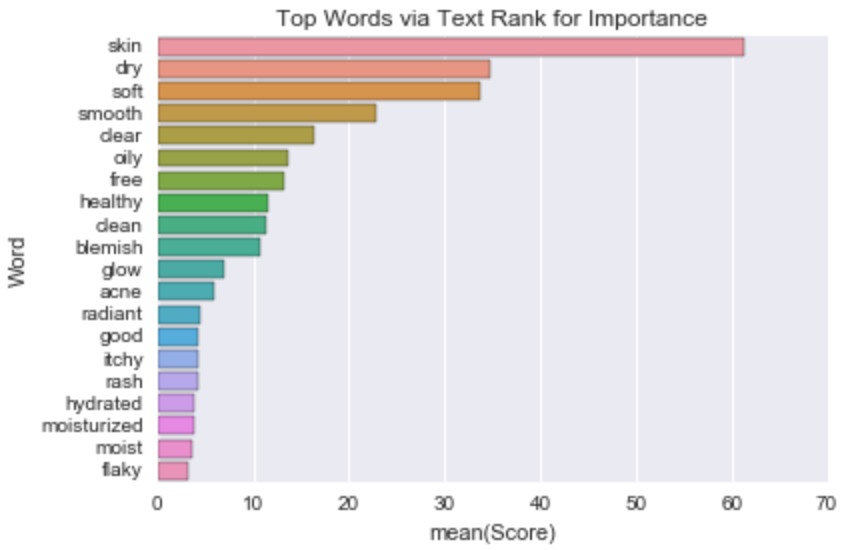
For the amazon.com product review dataset, these two methods generated vastly different results because the data set is huge and the reviews are in free-text form. So it was very obvious that the words returned by the graph-based textrank algorithms is much better in returning words that are actually relevant (see report 1 for details).

In contrast, for the Unilever focus group data set, the two algorithms generated similar top words. The reason is that this new data set is in question-answer format, which makes the texts very specific and to the point.

Even so, after cleaning the data set and experimenting with both methods, I still find textrank outperforming word count. Below is a demonstration of the results I get for both methods with the text from the first column 'What is Healthy Skin?'. For this column, 80% of the top 20 words are the same for both methods. The difference lies in the “difference” words. Let’s take a close look: the words that only appears in the result of the word count methods are “look”, “moisturize”, “feel”, “tone, while the words that only appears in the text rank method are “itchy”, “rash”, “moist”, “flaky”. So once again, the words that are only generated via textrank are much more relevant than those via word count in every case. So we decide to use the text rank results for grouping synonyms together in the following section.

Additionally, here is an explanation of why decide NOT to delete the nouns and only look at the adjectives: the top words generated by textrank contains nouns that are actually quite useful for our purpose - those nouns are very relevant.






## XII. Synsets generation/ grouping synonyms [by Shuni Fang]

To be able to not only return the top words, but also return the grouping of them in terms of synonyms, we first tried generating synonyms with WordNet. Because it is not a thesaurus for words in the context of beauty industry, the result is not satisfying.

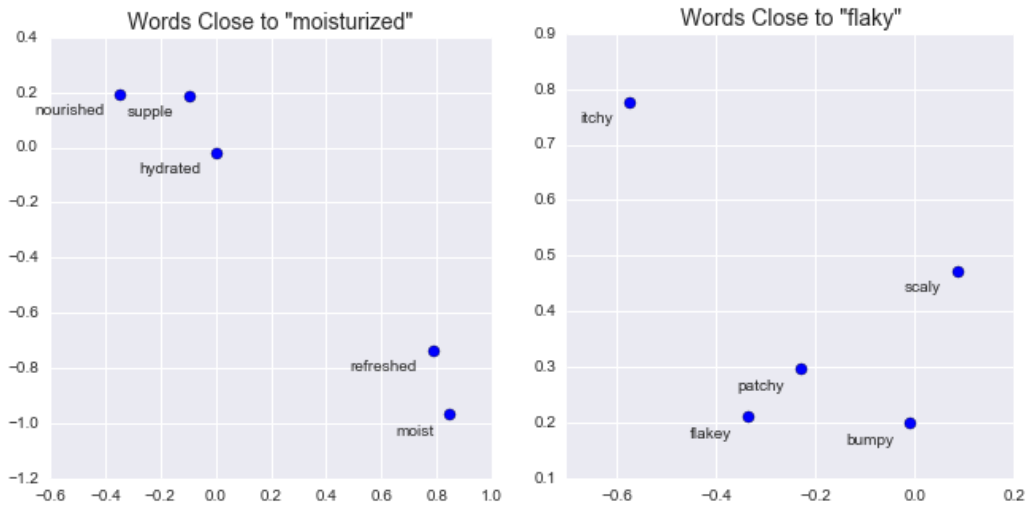
Then we consider using or training a Word2Vec similarly model in Gensim to get words that are close of each other. First we tried some of the famous pretrained model in Gensim, such as the famous GoogleNews-vectors-negative300.bin model, for the sake of save the training time, they didn't generate good result given that they are not specific to the skin-care or beauty industry.

So eventually we consulted the sense2vec model, which distinguished a word's different meaning when it is in different part of the text (with part-of-speech tags)[18]. We adopt the use Cython, which allows multi-thread processing, to use it with python to speed up the training process. When training the model, we found that the parameters in model training that have the most impact on the result are "min_count" and "size". Generally, given a big enough data set, the bigger the size, the better the result. So considering the size of the training data we have, we found that a size around 500 works well.

To get good grouping of synonyms, we want to be "strict" on "similarity". Using the model we trained with the whole amazon.com product review corpus, we find that the top five words are all quite relevant (ie. they are really synonyms in the context). To demonstrate, we plotted the five

[18] https://arxiv.org/pdf/1511.06388.pdf

synonyms of “moisturized” and “flaky” in scatter plot as below. The graphs give a direct view of the distance between words in the synsets:




We get good results by returning the top five most similar words of each key words. The table below shows top sample synonyms via Word2Vec for keywords from “What is healthy skin?”:

| Words | Top Five Synonyms via Word2Vec |
|---|---|
| moist | hydrated, moisturized, supple, nourished, refreshed |
| oily | greasy, acneprone, oilier, oilyand, oilybut |
| flaky | flakey, patchy, scaly, itchy, irritated |
| acne | breakouts, blemish, blemishes, rosacea, pimples |
| glow | sheen, dewy, glowy, brightness, glowing |

Overall, we find the results (ie. the top five “most similar” words via cosine distance of the word vectors) satisfying.

## XIII. Returning top words in groups of synsets [by Shuni Fang]

Now that we have the top words and the sets of synonyms, we are close to what we want but not yet there. Our final goal is to be able to return keywords in groups of synonyms. One obvious approach is to cluster the top words in groups of synonyms. The other easy-to-see approach is to generate a dictionary of synonyms of the words we care about. I personally found that the second approach generated quite good result. Below is the preliminary result for returning top words using text rank while grouping them into groups of synonyms via the dictionary method:

```
{'acne': ['blemish'],
 'blemish': ['acne'],
 'flaky': ['oily', 'itchy'],
 'hydrated': ['healthy', 'moisturized', 'moist'],
 'itchy': ['flaky'],
 'moist': ['hydrated', 'moisturized'],
 'moisturized': ['hydrated', 'moist'],
 'radiant': ['healthy'],
 'soft': ['smooth']}
```

We are still trying to figure out a better method of returning groups of top words either by graphs to text output. But as you can see from the preliminary results above, the dictionary method generate a good-enough result for grouping top words. And it is fast in doing it, which makes it possible for future effort in building a simple app to put into real-time usage.

## XIV. Word2Vec [by Junhui Liao]

In order to cluster key words for the responses from each survey question, I decide to train a Word2Vec[19] model on the Amazon product review[20] dataset. A sample review would look like the following:

*Very oily and creamy. Not at all what I expected… ordered this to try to highlight and contour and it just looked awful!!! Plus, took FOREVER to arrive.*

I choose to lowercase each word and remove numbers. In addition, it is better not to remove stop words since the algorithm relies on the context of the sentence in order to produce

[19] Tomas Mikolov (2013)
[20] Julian McAuley (2016)

high-quality word vectors. I utilize **gensim Word2Vec**[21] model and it expects single sentences as input, each one as a list of words. In other words, the input format is a list of lists. In order to tokenize into sentence level, I use NLTK's **punkt**[22] tokenizer for sentence splitting. A sample output would look like the following:

*[[very, oily, and creamy], [not at all what i expected ordered this to try to highlight and contour and it just looked awful], [plus, took forever to arrive], ...]*

There are a number of parameter choices that influence the quality and running time of the final model.

**Architecture:** Skip-gram (default) or continuous bag of words. I use skip-gram, which is recommended by the paper.

**Training algorithm:** Hierarchical softmax (default) or negative sampling. I choose softmax.

**Downsampling of frequent words:** Since the paper recommends values between .00001 and .001, I set this value to be .001.

**Word vector dimensionality:** I use 400 for this model because higher dimension can provide better results

**Context / window size:** I choose 10 for my model since most of the reviews are around 30 characters.

**Minimum word count:** I choose this value to be 40. It helps me to limit the size of my vocabulary so any word that does not occur at least this many times across all documents is ignored.

The following table summarizes my parameters setting:

| **Architecture** | **Training algorithm** | **Downsampling of frequent words** | **Word vector dimensionality** | **Context / window size** | **Minimum word count** |
|---|---|---|---|---|---|
| Skip-gram | Softmax | .001 | 400 | 10 | 40 |

Since our purpose is to  cluster key words for responses from each survey question, I want to see the cosine-similarity among frequently mentioned words. I pick up three frequently

[21] Radim Rehuren (2016)
[22] NLTK

mentioned adjective from question 1 and expect to see the other four highly mentioned adjective to show up as one of the top 10 cosine-similarity word. In other words, the union of cosine-similarity of selected words should cover my expected words. The ***most_similar*** function will be utilized to calculate the top 10 cosine-similarity words. I show question 1 as an example here. Please reference to **Appendix A** for the cosine-similarity value of the expected words.

**Question 1: What is Healthy Skin**

**Selected word: [acne, oily, smooth]**
**Expected word: [blemishes, soft, moisturized, dry]**
**Result: All show up in the most_similar() function**

From this example, I have all the expected words show up. In other words, these words are similar to each other under the Amazon Beauty review content and the survey response. Next section I discuss a naive approach to select keyword from the survey dataset and use the word embedding model above to cluster them.

## XV. Cluster [by Junhui Liao]

For clustering, I use K-Means[23]. The letter "K" represents the number of clusters to be set. K-Means clustering can be very slow if K is large but we may acquire better accuracy with larger K. Due to the computational and financial constraints, I set K to be 10 so that I have a reasonable running time as well as an acceptable outcome. I use the K-Means algorithm[24] from scikit with the following parameters setting

**N_cluster:** I set this value to be 10

**Max_iter:** 300 (Default) maximum number of iterations of the algorithm to run. I use the default value.

**N_init:** 10 (Default) I find the default value performs pretty well.

Our idea is to cluster similar words into the same group so that people can have a better better understanding of the text data. For keyword selection, I apply word count to similar words from responses and specify word type according to each question. Take question 1 as an example. I choose word count from high to low and specify the word type to be adjective because this type of words explain the problem better. The following table shows the question, word count as well as the cluster result.

[23] MacQueen, J. B. (1967)
[24] K-Means Algorithm

**Question 1: What is healthy skin**?
**word count table:**

| **Word:** | **moisturized** | **smooth** | **soft** |
|---|---|---|---|
| **Word Count:** | 63 | 107 | 122 |

**Selected word: [moisturized, smooth, soft]**
**Result:**

| **word:** | **moisturized** | **smooth** | **soft** |
|---|---|---|---|
| **Cluster/ Group:** | Group 5 | Group 5 | Group 5 |

The result is interesting because all the words have been categorized into Group 5. I also find out that the clustering result is consistent for question 1, 2, 3, 7 but not for question 4, 5. Please refer to **Appendix B** for the remaining results. The next step is to see direct keyword extraction from the survey response data because I are missing some keywords like, “not dry”, “not itchy”, and “not flaky”.

## XVI. Keywords selection [by Shengzhong Yin]

### Motivation

In the previous part, our logic is that we generate a set of interested words/phrases with respect to each question in the service data and then use the word vectors learned by the amazon.com data to detect synonyms, sentiments, non-relevant terms, etc. We first obtain a large pool of candidate keywords, then we filter them by result from embeddings. Originally we use simple word count to get the keywords. However,  we found that the keyword list obtained only by word counting is not satisfactory enough, because

1. The word counting output will always be single words, but our expecting keywords may contain phrases.
2. Words selected only by counts may not necessarily be representative.

### Interpreting the process

For these reasons we sought to find some other way to generating keyword lists. After exploring we chose to use RAKE[25] (Rapid Automatic Keyword Extraction) in python. RAKE is a simple keyword extraction library which focuses on finding multi-word phrases containing frequent words[26]. To use RAKE, we first need to give it a stop words list. This can be obtained from

[25] https://github.com/zelandiya/RAKE-tutorial
[26] https://www.airpair.com/nlp/keyword-extraction-tutorial

NLTK's stopwords, or RAKE's self-contained stopwords file. We chose the second. Also three parameters could be specified:

1. The least number of characters a word may have
2. The largest number of words a phrase may contain
3. The least number of times a keyword appears in the text

Given all the parameters, RAKE first splits the text (after lemmatization) into sentences and generates the candidates according to the words listed in the stopwords file. In this case candidate keywords consist of more than one words might be maintained, for example, since "blemish" is not a stop word, phrases like "blemish free" could stay in the candidate pool.

Second, RAKE computes the score of each candidate based on its inner criterion. The detail is not shown publicly but the criterion is described in doc as a word is scored "according to their frequency and the typical length of a candidate phrase in which they appear". A phrase' score will simply be the sum of the scores for each of its words.

Finally RAKE ranks the candidates based on property scores from the previous step. The following is the output of the first question "What is Healthy Skin?", if we set the parameters to be 5, 3, 3 (that means, each word in phrase must contain at least 5 characters, each phrase must contain at most 3 words and each keyword must appear at least 5 times) and we want to see all phrases with score >= 2:

[('blemish free', 4.266666666666667),
('skin tone', 4.204545454545455),
('overly dry', 4.111111111111111),
('dry patches', 3.888888888888889),
('feels soft', 3.873194221508828),
('dry spots', 3.861111111111111),
('feels good', 3.8240870786516856),
('good color', 3.8159722222222223),
('break outs', 3.666666666666667),
('healthy skin', 3.6452234206471497),
('healthy glow', 3.4406779661016946),
('feels smooth', 3.1933000416146484)]

## Tuning the parameters and conclusion

The above list of keywords somewhat makes sense because the phrases did describe things about how people may consider to be properties of healthy skins. RAKE also has a script evaluating accuracy, recall and F1 score based on manually assigned keywords but unfortunately our mentor did not provide us that as promised.

So the only thing I could do at this stage is to tune the parameters manually. I did the selection on two criterions:

1. The numeric scores calculated by RAKE for each keyword
2. The correlation each keyword has with respect to the question based on my subjective view

The parameters I selected are just 5, 3, 3 as before. The first parameter (the least number of characters a word may have) is trivially set to be 5 to avoid words like “I”, “a”, “the”, etc. The second parameter (the largest number of words a phrase may contain) does not affect the output much in our case because our text does not contain many high-frequency long phrases. The third parameter (the least number of times a keyword appears in the text) makes the most difference here. The service data is relatively small so some representative phrases do not appear much in the text, for example, if I set the third parameter to be 5 and limit the scores to be >= 1.5, the output will look like:

[('feels soft', 3.926262626262626),
('feels good', 3.8381944444444445),
('good color', 3.8159722222222223),
('healthy skin', 3.6452234206471497),
('feels smooth', 3.2074074074074073),
('feels', 1.7444444444444445),
('color', 1.7222222222222223),
('wrinkles', 1.5)]

We see the keywords are much broader and the scores becomes low. Moreover some representative phrases (“blemish free”, “dry patches”, etc) are not in the list anymore. So I set the final parameters to be 5, 3, 3. As a conclusion, we believe RAKE is a good way to produce candidate pool of keywords for summarizations.

## XVII. Extractive Summarization [by Joshua Safyan]

### Introduction

The extractive summarization task seeks a subset of source sentences which best convey the essential information of the original text. In the *multi-document* summarization setting, these sentences come from a corpus of related documents. Both the Amazon product review dataset and the Unilever focus group response data can be viewed as instances of the *multi-document* summarization regime. For a given product in the former, each review is an individual document.

For each survey question in the latter, each response is a separate "document." In both cases we consider individual sentences to be the basic textual unit, following the literature[27] [28].

We implement extractive summarization techniques from two papers: "Multi-document Summarization via Budgeted Maximization of Submodular Functions" (Lin and Bilmes, 2010) and "Extractive Summarization by Maximizing Semantic Volume" (Dani Yogatama et al. 2015). The remainder of this section describes the techniques, related research, our results, and directions for future inquiry.

## Multi-document Summarization via Budgeted Maximization of Submodular Functions (Lin and Bilmes, 2010)

### Submodularity

A submodular set function is one that exhibits the property of decreasing marginal returns. That is, as a set gets larger, the addition of a marginal element to that set results in a smaller marginal increase in the value of the function. Submodularity can also be thought of as a discrete analog to convex and concave functions[29]. More formally:

> *"Given a set X, a set function* $f : 2^X \rightarrow \mathbb{R}$ *is called submodular if for any two sets* $S1$ *and* $S2$ *such that* $S1 \subset S2 \subset X$ *and element* $x \in X \setminus S2, f(S1 \cup \{x\}) - f(S1) \geq f(S2 \cup \{x\}) - f(S2)$*"* [30]

While one can minimize submodular functions in polynomial time, the submodular maximization problem is NP-complete[31]. Thus, in practice greedy algorithms are used to approximate optimal solutions. Lin and Bilmes(2010)[32] frame the extractive summarization task as a submodular maximization problem subject to a budget (summary length) and propose a greedy algorithm to solve it. In the case of extractive summarization, the summary S is a subset of sentences from the complete set of sentences V. Adding the most relevant sentences in order to the summary S, the marginal contribution of the first several sentences is much greater intuitively than that of the nth sentence. While the marginal contribution of the N-1th sentence will still be greater than or equal to zero, summaries should also be parsimonious. This brevity condition is achieved by setting a budget for the optimization problem (the length of the summary) and penalizing redundant sentence selections for the summary subset.

---

[27] Lin and Bilmes, 2010 http://melodi.ee.washington.edu/people/hlin/papers/naaclhlt2010.pdf
[28] Dani Yogatama et al. 2015 http://www.cs.cmu.edu/~dyogatam/papers/yogatama+liu+smith.emnlp2015.pdf
[29] ibid.
[30] Hayato Kobayashi et al. 2015 http://www.aclweb.org/anthology/D15-1232
[31] http://melodi.ee.washington.edu/people/hlin/papers/naaclhlt2010.pdf
[32] ibid.

### Formulation of the submodular objective function

Lin and Bilmes (2010) formulate the extractive multi-document summarization task as a budgeted optimization problem over a weighted, undirected graph G, where nodes are the tf-idf sentence vectors, and edges are the cosine distances between them. Building on the work of (Carbonell and Goldstein, 1998), they use a form of the *Maximum Marginal Relevance* (MMR)[33] score function.

$$f_{MMR}(S) = \sum_{i \in V \setminus S} \sum_{j \in S} w_{i,j} - \lambda \sum_{i,j \in S: i \neq j} w_{i,j}, \lambda \geq 0$$

$V \setminus S$ is the set of all sentences $V$ less the sentences in $S$, the set of sentences in the summary. $w_{i,j}$ is the edge from node i to node j, and $\lambda$ is a penalty term on internal similarity of the summary. The first term seeks to maximize the similarity between a subset of sentences $S$ (the summary) and the remaining sentences in the source $V \setminus S$. Since we seek to maximize the first term, which is the graph cut function, the task of finding a maximally representative subset of sentences can also be seen as the Max-Cut of the graph G, which is NP-complete[34].

$$f_{cut}(S) = \sum_{i \in V \setminus S} \sum_{j \in S} w_{i,j}$$

The second term penalizes the summary for redundancy; higher values of $\lambda$ induce the summary to be more diverse, while lower values of $\lambda$ make the summary more closely represent the majority of the remaining sentences regardless of repetition. We also pick a budget $B$ which is the maximum length of the summary as measured in characters, words, bytes, etc.

While one can write an Integer Linear Program (ILP) to optimally solve the objective function subject to the budget constraint, evaluating the ILP is prohibitively time consuming for even modestly sized documents and summary sizes. Instead, Lin and Bilmes propose a greedy polynomial time algorithm to approximately maximize the objective function. Furthermore, they prove that it attains a result within 63% of the optimal solution[35], with empirical results tending to be nearly optimal.

*Modified greedy algorithm*[36]

$1: \ G \leftarrow \varnothing$

$2: \ U \leftarrow V$

$3: \ \textit{while } U \neq \varnothing \textit{ do}:$

$4: \quad k \leftarrow \arg\max_{\ell \in U} \left(f(G \cup \{\ell\}) - f(G)\right) / (c_l^r)$

$5: \quad G \leftarrow G \cup \{k\} \textit{ if } \sum i \in G \, c_i + c_k \leq B \textit{ and } f(G \cup \{k\}) - f(G) \geq 0$

---

[33] http://www.cs.cmu.edu/~jgc/publication/The_Use_MMR_Diversity_Based_LTMIR_1998.pdf

[34] Karp 1972 https://people.eecs.berkeley.edu/~luca/cs172/karp.pdf

[35] http://melodi.ee.washington.edu/people/hlin/papers/naaclhlt2010.pdf

[36] ibid.

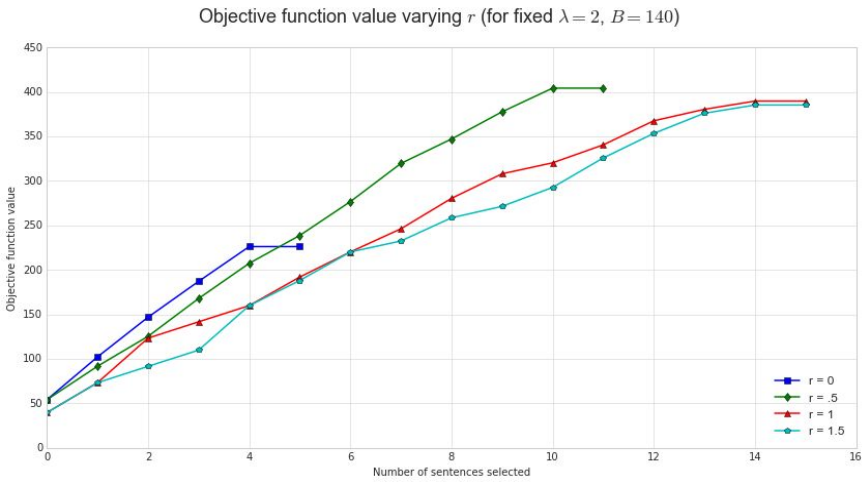
$6: \quad U \leftarrow U \setminus \{k\}$

$7:$ *end while*

$8: v^* \leftarrow argmax\ v \in V, c_v \leq B * f(\{v\})$

$9:$ *return* $G_f = arg\ max\ S \in \{\{v^*\}, G\}\ f(S)$

### Modified Greedy Algorithm Results




Tuning the parameter r for the scaled cost. r < 1 biases toward longer sentences, while r > 1 biases toward shorter sentences

The modified greedy algorithm performed fairly well on both the Amazon and Unilever datasets by manual inspection. The MMR objective function score can be seen as an alternative to ROUGE[37], allowing us to do a grid-search experiment with varying values of $\lambda$ and $r$ for a for fixed summary length. While tuning the parameters of the algorithm allows the user flexibility in the type of summary produced, we found that fairly different parameters were needed for the Amazon and Unilever datasets to produce informative, non-redundant summaries. Please see appendix C for samples.

### MMR and Embeddings

While the modified greedy algorithm performs well as a baseline, it is still computationally expensive, requires parameter tuning, and fails to account for semantic information in the sentence vectors. Follow-on work has since sought to address these issues. Kageback et al.[38] proposed a modification to the greedy MMR algorithm to add semantic information to the summaries by summing word embeddings to form continuous space sentence vectors instead of tf-idf sentence vectors.

However, constructing sentence vectors by adding together word embeddings still takes somewhat of a bag-of-words approach in that it disregards word order. Kobayashi et al. address

[37] ibid.

[38] http://www.aclweb.org/anthology/W14-1504

this assumption by considering whether document vectors could instead be used in the submodular optimization setting and prove that the objective function with cosine similarity of document vectors is not submodular[39]. They instead consider a similarity measure related to the distribution of word embeddings which has connections to KL-divergence and earth mover's distance (EMD).

### Extractive Summarization by Maximizing Semantic Volume (Dani Yogatama et al. 2015).

As mentioned above, more recent work has focused on adding semantic information to the summary sentence selection process. Yogatama et al. introduce *semantic volume*, a new score function related to coverage-based summarization. The authors consider continuous vector representations of sentences, specifically sentence embeddings, which together span a high dimensional semantic space. To capture as much information as possible, the embeddings of the sentences selected for the summary should form a convex hull of this semantic manifold. However, as the authors note, a convex hull might not be possible given length constraints on summaries. They therefore posit that the summary sentences should span as large of a subspace as possible, i.e., maximize the semantic volume[40].

Following Yogatama et al., we construct a document-term matrix. While Yogatama et al. use term frequency for bigrams (removing bigrams consisting only of stop words), we use 1,2-grams with tf-idf weighting due to the smaller Unilever data set. Then, following the paper, we apply Singular Value Decomposition and take the $N x K$ matrix of left singular vectors (this is the same technique as LSA). This creates a lower dimensional approximation of the $N x D$ matrix, where $K < D$. The paper suggests $K$ of 500 to 600 worked well, but we had to use even lower dimensions due to the small size of the Unilever data. The recommended reduced dimensionality from Yogatama et al. exceeded the original dimensionality of the Unilever document term matrix itself. Nevertheless, reasonable results were achieved with $K = 100$.

**Algorithm 1** Greedy algorithm for approximately maximizing the semantic volume given a budget constraint.

**Input:** Budget constraint $L$, sentence representations $\mathcal{R} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N\}$
$\mathcal{S} = \{\}, \mathcal{B} = \{\}$
Compute the cluster centroid $\mathbf{c}$: $\frac{1}{N}\sum_{i=1}^{N}\mathbf{u}_i$.
$p \leftarrow$ index of sentence that is farthest from $\mathbf{c}$.
$\mathcal{S} = \mathcal{S} \cup \{s_p\}$. ▶ add first sentence
$q \leftarrow$ index of sentence that is farthest from $\mathbf{s}_p$.
$\mathcal{S} = \mathcal{S} \cup \{s_q\}$. ▶ add second sentence
$\mathbf{b}_0 = \frac{\mathbf{u}_q}{\|\mathbf{u}_q\|}, \mathcal{B} = \mathcal{B} \cup \{\mathbf{u}_0\}$
total length $=$ length$(s_p)$ + length$(s_q)$
**for** $i = 1, \ldots, N-2$ **do**
  $r \leftarrow$ index of sentence that is farthest from the subspace of Span$(\mathcal{B})$. ▶ see text
  **if** total length + length$(s_r) \leq L$ **then**
    $\mathcal{S} = \mathcal{S} \cup \{s_r\}$.
    $\mathbf{b}_r = \frac{\mathbf{u}_r}{\|\mathbf{u}_r\|}, \mathcal{B} = \mathcal{B} \cup \{\mathbf{b}_r\}$.
    total length $=$ total length + length$(s_r)$
  **end if**
**end for**

[39] http://www.aclweb.org/anthology/D15-1232
[40] http://www.cs.cmu.edu/~dyogatam/papers/yogatama+liu+smith.emnlp2015.pdf

### Semantic Volume Maximization Results

The greedy algorithm for approximate semantic volume maximization appeared to provide good summaries (summaries were taken to be 300-500 characters). It performed well on both the Amazon and Unilever data with no modifications between data sets. Please see appendix C for samples.

### Comparison to MMR

Both MMR and semantic volume maximization are budgeted techniques. Each has as input a budget B, which is the maximum length of the summary (in words, characters, bytes, etc). While MMR has the term $\lambda$ to penalize internal summary similarity, the semantic volume score automatically finds diverse sentences and has the advantage of not needing to be tuned for different data sets. Additionally, the semantic volume algorithm is very fast. For a set of 407 source sentences, it generates a 300 character summary in slightly under 100ms on a 1.3Ghz Core M processor with 8GB of RAM (compared to approximately 10 seconds for MMR).

On the other hand, one might prefer MMR if trying to find strictly the most prevalent information from the source documents. By tuning $\lambda$ toward zero, the summary will closely resemble the remaining text, while semantic volume explicitly tries to cover a wide range of information. Thus, the modified greedy algorithm for MMR has some additional flexibility with regard to redundancy and sentence length compared to semantic volume maximization.

## XVIII. Future Work

Both the greedy submodular optimization and volume maximization techniques can benefit from richer vector representations of sentences, as each rely on distances between vector representations of sentences. In fact, Yogatama et al.[42] note that their greedy algorithm for semantic volume maximization could be used as a benchmark for comparing different sentence embedding strategies.

One approach to more robust text representations might be the neural autoencoder[43]. For shallow networks, autoencoders find a lower dimensional representation similar to PCA. With deeper networks, however, an autoencoder can learn nonlinearities that allow it to create a more robust low-dimensional representation of an input.

---

[41] ibid.
[42] ibid.
[43] Jiwei Li et al. http://web.stanford.edu/~jurafsky/pubs/P15-1107.pdf

Appendix A: [by Junhui Liao]

**model.most_similar(“acne”)**
[('breakouts', 0.7275450229644775),
 ('blemish', 0.7041106224060059),
 ('rosacea', 0.659509539604187),
 ('pimples', 0.6555451154708862),
 ('blemishes', 0.6529945135116577),
 ('breakout', 0.610044538974762),
 ('acnes', 0.6019399166107178),
 ('eczema', 0.5939148664474487),
 ('melasma', 0.5863379240036011),
 ('zits', 0.5698316693305969)]

**model.most_similar(“oily”)**
[('greasy', 0.6574270725250244),
 ('oilier', 0.5255621671676636),
 ('flaky', 0.5210124850273132),
 ('dry', 0.48807603120803833),
 ('itchy', 0.47405949234962463),
 ('irritated', 0.4707390069961548),
 ('sticky', 0.47032323479652405),
 ('slick', 0.4661816656589508),
 ('dehydrated', 0.46498042345046997),
 ('sensitive', 0.45952892303466797)]

**model.most_similar(“smooth”)**
[('smoothe', 0.6627895832061768),
 ('soft', 0.5832105278968811),
 ('smoothly', 0.5729053616523743),
 ('silky', 0.5427254438400269),
 ('moisturized', 0.5364487171173096),
 ('hydrated', 0.518983781337738),
 ('supple', 0.513997495174408),
 ('shiny', 0.5126349925994873),
 ('smooths', 0.5077080130577087),
 ('nourished', 0.4913996458053589)]

## Appendix B: [by Junhui Liao]

**Question 2: How you know your skin is healthy?**
**Selected word: [clear, good, healthy]**
**Result:**

| **Clear** | **Good** | **Health** |
|---|---|---|
| Group 4 | Group 5 | Group 5 |

**Question 3: How do you know your skin is getting healthier with every shower?**
**Selected word: [clean, moisturized, soft]**
**Result:**

| **Clean** | **Soft** | **Moisturized** |
|---|---|---|
| Group 5 | Group 5 | Group 5 |

**Question 4: How do you get healthy skin?**
**Selected word: [care, eat, lotion, drink]**
**Result:**

| **Care** | **Lotion** | **Drink** | **Eat** |
|---|---|---|---|
| Group 4 | Group 4 | Group 6 | Group 8 |

**Question 5 How do you maintain healthy skin?**
**Selected word: [moisturize, eat, drink, clean]**
**Result:**

| **Clean** | **Drink** | **Eat** | **Moisturized** |
|---|---|---|---|
| Group 5 | Group 6 | Group 8 | Group 5 |

**Question 7 How bar soap or body wash gives you healthy skin?**
**Selected word: [clean, ingredients, soap]**
**Result:**

| **Ingredients** | **Soap** | **Clean** |
|---|---|---|
| Group 2 | Group 2 | Group 5 |

## Appendix C: [by Joshua Safyan]

### Comparison of greedy algorithms for MMR and semantic volume maximization

Summary budget is 300 characters for each.

| Question | Greedy Algorithm for Budgeted Submodular Optimization | Greedy Algorithm for Semantic Volume Maximization |
|---|---|---|
| ***What is Healthy Skin?*** | smooth. not dry not oily. dry skin. good elasticity, moist no age spots. soft, not scaly. glowing moisturizes skin. clear, not dry, not oily. smooth. not dry or oily. moisturized, healthy color, no blemishes. free of blemishes, shiny, not dry or oily. not dry, no acne.. not dry. what. skin that is not dry.. clean. not dry | no scals or cracks. no flakes. vibrant, clear & moisturized. no irritations. shinny, smooth. smoothness, no rash, no itch. not too dry or oily, no acne. it's hydrated and elastic.. .not itchy... not.dry supple soft. it is clean. even toned. glowing skin that is dirt-free. hydrated. |
| ***How do you know your skin is healthy?*** | looks and feels good. it's not itchy or dry or dull. its not flaky. by looking at it. if it is firm and glowing with out a whitish tint from dryness. feel it. no marks. by the way it looks. normal not dry. smooth, soft. look/feel. its not. how it looks. very shiny. my skin feels moisturized. no recent breakouts or clogged pores | i prefer to have some smoothness and some softness to the touch. appearance. not scaly or flaky. it is soft, not chapped,.. again, not to dry not to oily, a glow??. snaps back, not dry or itchy. when i am comfortable. by how it looks and feels... |
| ***How do you know your skin is getting healthier with every shower?*** | i dont. i dont. my pores care closing. skin stays soft. the way it feels.. i dont. depends on how dry u r after. i dont. cleaner. flakiness disappears and its smooth. glows. by the feel of it. not sure.. products. it feels good. less flakes. softer. softness. i dont. not dry. feel. look vibrant. moisturized, clear and clean feeling. soft | softer and smoother looking. moisture, no flakes. i'm sure some products are more natural and nourishing than others.. don't know. i have no idea. .. glows. i dont. i can see suttle improvements.. not dry. i dont!. dryness after. by the appearance. by the way it feels. i don't. |
| ***How do you get healthy skin?*** | moisture. healthy diet. good cream. moisturie. cleansing, drink water. moisture. daily care. taking care of your face. daily care. hydrate. moisturize. take care of your skin inside and out. by washing it and using cream. using correct products. maybe medication, using the right products, maybe without perfume or dyes. take care | by taking care of it. by using moisturizing products and drinking lots of water. exfoliate. lubricating the skin consistently. dont know. cream, moistureizer. by using lotions, shaving. |
| ***How do you maintain healthy skin?*** | lotion. cleaning it daily. drinking lots of water. lotion. good diet, avoid smoking. diet. moisture. conditioning the skin. moisturize daily. mosturize. stay moisturized, cleansed, use good products. products. mosturize it. moisturize daily. lotion. keep up with your regimen. as described earlier. continue to moisturize. healthy diet | drink water and moisturize. keep it clean and moisturized. vitamins, lotions. washing and moisturiing. use products for your skin type daily. eating health. it's about consistency. same as last answer. see previous question a healthy diet and the correct products. constant routine. |